Homework Assignment 3

ECE438

Due 5/12/2020

1. **(15 points)** For the following problem, assume a 5-stage pipelined processor with forwarding and hardware interlocking. Also, assume branch resolution is in the Execute stage. Consider the code below:

```
    addi $t2, $t1, 60
loop:
    lw $t4, 0($t1)
    lw $t5, 4($t1)
    xor $t6, $t4, $t5
    sw $t6, 8($t1)
    addi $t1, $t1, 12
    bne $t1, $t2, loop
```

   (a) How many loop iterations does the above code execute?

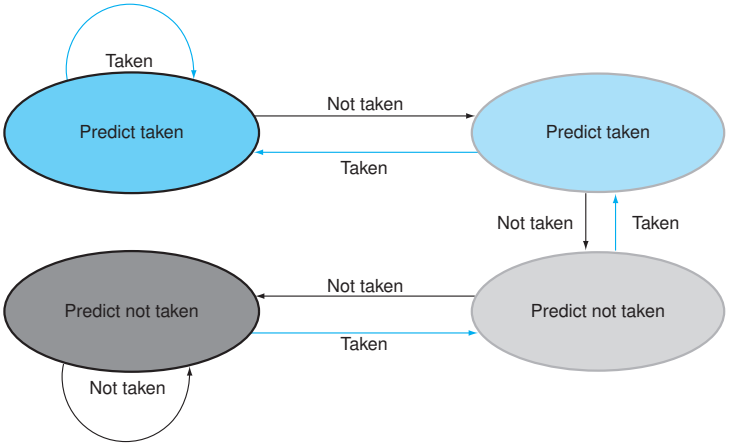
   (b) Identify the data dependencies in the above code.

   (c) Draw the pipeline execution diagram for the first *two* iterations of the above code when an "assume not taken" branching scheme without a branch delay slot is used.

   (d) How many clock cycles are required to execute the above code to completion when an "assume not taken" branching scheme without a branch delay slot is used?

   (e) Modify the code to take advantage of a branch delay slot. How many clock cycles are required to executed your modified code to completion when an "assume not taken" branching scheme with a branch delay slot is used?

   (f) How many clock cycles are required to execute you modified code assuming a 100% correct branch predictor in the decode stage in addition to the branch delay slot.



   (g) Consider the use of the two-bit predictor shown above in the decode stage. Assuming the predictor starts in the top right state, how many clock cycles are required to execute your code?

(h) What is the accuracy of this predictor given the code above? Compare that to the accuracy of the "assume not taken" scheme.

(i) What speedup does the branch predictor from Problem 1g provide over the "assume not taken" scheme from Problem 1d?

(j) Further rearrange the code to reduce the number of stalls due to data dependencies.

2. **(30 points)** The following problems are related to techniques and terminology of the memory hierarchy.

(a) Define spacial and temporal locality. Describe the role spacial and temporal locality play in the memory hierarchy.

(b) Define and briefly describe the types of cache misses talked about in lecture. Remember the three C's.

(c) Draw the hardware diagram of a *direct-mapped cache*. Be sure to label the various portions of the address.

(d) Compare and contrast *write-back* and *write-through* cache write policies. When might one be preferred over the other?

(e) Compare and contrast *write-allocate* and *no write-allocate*. When might one be preferred over the other? Can either of these schemes be used with *write-back* and *write-through*? Why or why not?

(f) Describe the purpose of a *write buffer*. Where in the memory hierarchy might you place a write buffer? Draw a diagram showing where you might place one and explain your reasoning.

(g) What is the purpose of increasing the block size of a cache? What happens if the block size is too large?

(h) Why might one add *associativity* to a cache? How might one add associativity? Draw a hardware diagram showing how to construct a 2-way set associative cache. What are the drawbacks to set-associative caches?

(i) What is a *split cache* scheme? Why is it beneficial? What might be a disadvantage? Draw a diagram of a single-level, split cache scheme.

3. **(15 points)** For the problem below, a direct-mapped cache is provided the following sequence of *word* addresses: 3, 4, 5, 6, 7, 1, 1, 3, 36, 39, 35, 36

   (a) Assuming an initially empty, direct-mapped cache with 16 one-word blocks, complete the table below, identifying the tag and index for each word address. Also, indicate whether the access was a hit or miss, and if a it was a miss indicate the type of miss (i.e. compulsory or conflict).

Reference	Address	Tag	Index	Hit/Miss	Type of Miss
3	00000011				
4	00000100				
5					
6					
7					
1					
1					
3					
36					
39					
35					
36					

   (b) Calculate the miss rate of the cache above. What is the hit rate?

(c) Complete the table below assuming a direct-mapped cache with 8 two-word blocks.

Reference	Address	Tag	Index	Hit/Miss	Type of Miss
3					
4					
5					
6					
7					
1					
1					
3					
36					
39					
35					
36					

(d) Calculate the miss rate of the cache above. What is the hit rate?

(e) Compare the miss rate in part (d) with that of part (b). Does it improve? If so, why?

(f) What two principles make memory caching effective? Be specific in your answer and provide a brief explanation of each principle.

4. **(10 points)** Assuming a direct-mapped cache with a byte-address that is broken up such that bits 0-2 are for the byte offset, bits 3-7 are for the word offset, bits 8-14 are for the index, and bits 15-31 are for the tag, answer the following questions:

(a) How large are the words in this machine?

(b) How many words are in each cache block? How many bytes are in each cache block?

(c) How many cache blocks are in the cache? How many sets are in the cache?

(d) How large is the data store of this cache?

(e) How large is the tag store? Assume a *valid* and *dirty* bit are included with each tag.

(f) If you were to modify the cache to be 2-way set associative but keep the data store the same size, what size would the tag and index be? How large would the tag store be?

(g) How much memory can the above machine address?

5. **(10 points)** Imagine you have a 1GHz RISC processor with split L1 instruction and data caches, each 32kB in size with a hit time of 1ns. Access to main memory takes 30 ns, and 38% of instructions access memory. The L1 instruction cache miss rate is 0.7%, while the L1 data cache miss rate is 6%.

   (a) Calculate the Average Memory Access Time (AMAT) for each of the L1 caches.

   (b) Assuming your processor has a CPI of 1.2 with an ideal memory hierarchy, what is the CPI considering memory stalls?

   (c) You are considering the inclusion of a 128kB L2 cache to improve your performance. If the miss rate of the L2 is 30%, what would the AMAT of the L1 instruction and data caches be with the L2 cache? Assume a 5ns L2 hit time.

   (d) What would the CPI of your processor be with the L2 cache? What is the speedup due to the L2 cache?

   (e) Why do you suppose the miss rate of the L2 is so much higher than that of the L1?

   (f) Why is the miss rate of the instruction cache lower than the miss rate of the data cache?